Predição de Desvios

Branch prediction são conjuntos de técnicas implementadas em hardware ou software, que têm o objetivo de reduzir os conflitos de controle em processadores de arquitetura Pipeline, através da previsão correta de que os desvios condicionais vão desviar ou não, reduzindo atrasos no fluxo de instruções posteriores aos desvios, conhecidos como bolhas no pipeline.

Apesar dos conflitos de controle serem menores que os conflitos de dados, quase 20% das instruções são de desvio condicional, tornando-se crítico alternativas para amenizar os gargalos, principalmente quando há muitas instruções no mesmo ciclo de clock. Para isto, existem as chamadas técnicas de execução especulativa que, com base em probabilidades de ocorrência ou estratégias, permitem que haja uma previsão se haverá desvio ou não, ou até mesmo trocar de ordem algumas instruções, sem alterar o resultado final do programa, para quando surgir uma condição de desvio, a mesma seja resolvida com o mínimo possível de bolhas no Pipeline. O conjunto de técnicas dividem-se em técnicas de software ou técnicas de hardware [1]

Técnicas de software

Em geral, neste tipo de técnica o hardware, após constatar que houve um desvio condicional, pausa a instrução de desvio e delega ao software, durante a compilação, reorganizar quais das instruções serão antecipadas à instrução de desvio. Assume-se que todas as técnicas de software são estáticas, uma vez que a estrutura não é alterada no próprio código, e sim virtualmente pelo compilador. Algumas das principais técnicas são:

Delayed Branch: desde que não haja Relações de dependência entre as instruções, o compilador tenta organizá-las da maneira mais adequada para reduzir as bolhas no pipeline. Consiste em antecipar as instruções que estão antes do desvio. Compiladores conseguem preencher com instruções até 50% dos espaços após o desvio.

Fetch Taken and Not-Taken Path: técnica difere de *Delayed Branch* no fato de que, em vez de adiar instruções que estão antes do desvio condicional, executar também instruções que estão após o desvio, desde que não haja problemas de dependências. Executa as duas possibilidades, do desvio acontecer ou não, os dois fluxos ao mesmo tempo. Não tem implementações pois a complexidade de hardware e custo seriam muito grandes.

Branch Folding(desvios de ciclo zero): usada em certas arquiteturas que possuem instruções extras chamadas *flags* de condição, para identificar o resultado do desvio no momento em que o mesmo é encontrado, através de comparação entre registradores ou operações aritméticas, fazendo com que o fluxo de controle seja alterado pelo compilador, caso necessário, a tempo de evitar ciclos desperdiçados. Processadores CRISP(AT&T), PowerPC 601 e IBM RISC System 6000 são exemplos de arquiteturas com branch folding.

In-line: quando as instruções de desvios estão dentro de chamada ou de retorno de funções torna-se difícil encontrar uma previsão eficiente, pois podem ser chamadas de diferentes pontos do programa. A técnica in-line consiste em substituir as chamadas de funções pelo conteúdo específico de cada uma, através da decodificação, onde é possível saber o conteúdo específico de cada instrução, onde o compilador, em caso de instruções redundantes, o procedimento de passagem de parâmetros não precise ser executado a cada chamada para a função, diminuindo o tempo de execução das instruções, apesar de que o código tem seu tamanho aumentado.

Desenrolamento de loops: técnica utilizada em instruções de desvios condicionais que estão dentro de um laço de repetição, trocando o laço quando possível pelo próprio conteúdo existente nele (código-objeto). Informações do conteúdo interno dos laços e número de repetições, fornecidos pela decodificação e unidade de controle, permitem redução de tempo de processamento, eliminando instruções de *load/store* para saber qual conteúdo está nos registradores.